The Star Method

Komi Agbo, Dalton Burke, Nick Mako, James Vance

University of Colorado Denver

Fall 2019

Overview

Sam's Hauling

What do they do?

- Provide roll off dumpsters of various sizes (cans)
- Customers
  - Delivery
  - Pickup
  - Switch

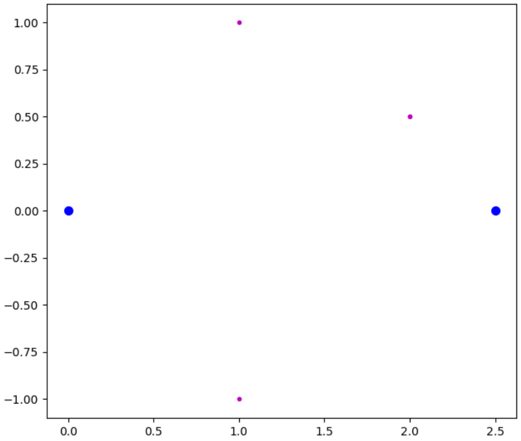
Sam's Hauling

What do they do?

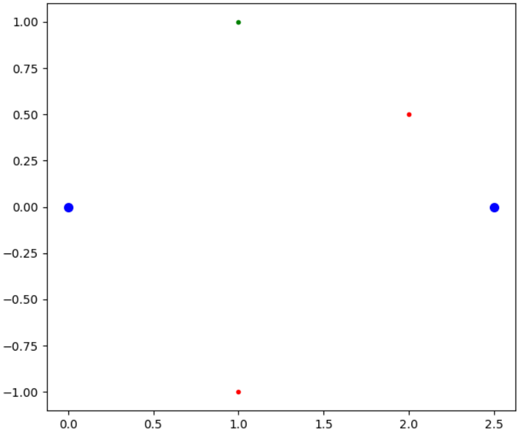
- Provide roll off dumpsters of various sizes (cans)
- Customers
  - Delivery
  - Pickup
  - Switch

And how do they do it?

- Resources
  - Multiple Trucks
  - Multiple Landfills
  - Multiple Storage Yards
  - Cans
- Constraints
  - A truck can hold only one can
  - Truck/Can size compatibility
  - Customer time preference (AM/PM/None)



Landfills (blue), Switches (magenta)

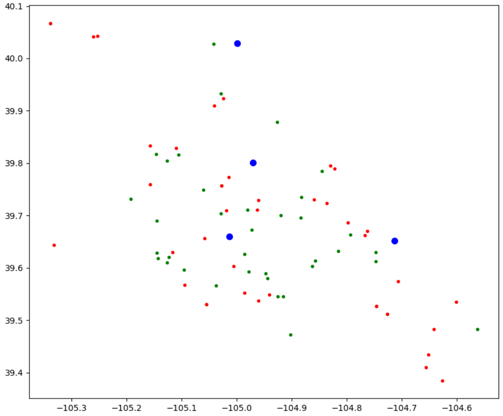


Deliveries (green), Pickups (red)

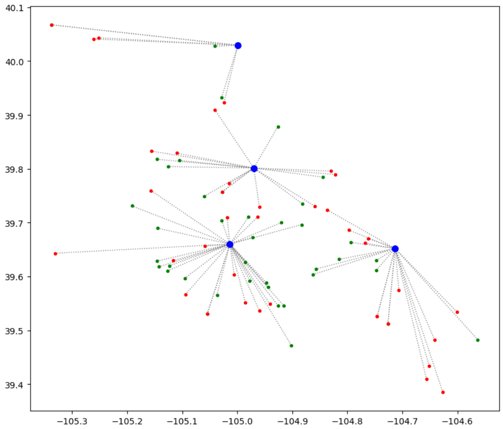
How do we find the best triangles?

It turns out this problem is very fast to solve, known as a matching problem, and the result is a truly optimal answer.

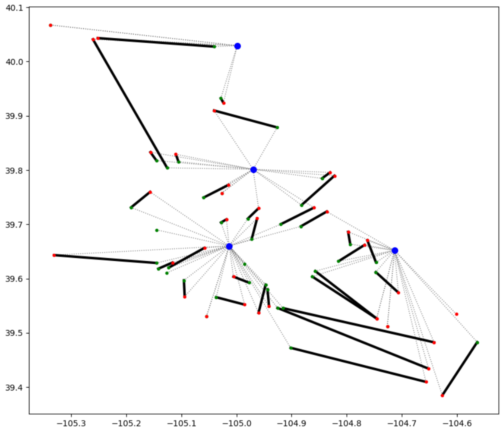
In our implementation we used the Munkres library, which has an implementation of the Hungarian Algorithm. This completes the triangle computation in $O(n^3)$ time, the heaviest lifting that we have to do.



Data from a recent day's work



Identify the nearest landfill to create *stars*



Identify the best set of pairs to create *triangles*

Now to actually build the routes for the drivers...

The algorithm for assigning the full schedule are as follows:

1. Receive data for the day and process it into the program.
2. Create pairs with Hungarian algorithm.
3. Assign driver to resolve truck size constraint.
4. Transition to the farthest zone that requires the truck resolving a hard can deficit if applicable.
5. Service all routes within the zone that require the truck and return to 2 if more applicable routes exist in other zones.
6. Finish route as per a normal driver and return to 1 if other truck size constraints exist.
7. Assign next driver to farthest zone with routes.
8. Transition to the zone.
9. If applicable resolve hard can deficits.
10. Assign remaining routes by prioritizing AM/PM constraints and then distance to the main hub.
11. If there are no more routes in the zone return to 6 choosing the next zone with routes remaining.
12. If the driver schedule is full including the transition back to the main

From an old schedule provided by the client, we ran our algorithm to see how it stacked up to the clients solution **(comparison pending, will be present in final)**. In the following figures, each color is a driver's route.

New requirements for the customer (file organization)

There are two things that would make this program better, one, is better data. If given actual driving times, the output would do a much better job of reflecting real world conditions. As of now, we actually can't even test how well it's doing.
The other thing that would make this better is a better pair choice reflecting the transitions that will inevitably need to be taken. Right now, pairs are chosen as if drivers can teleport between landfills. While a vast majority of a drivers time is *not* taking transitions, it still seems like we are losing a bit here. In order to truely take them into account however, it seems like a much more painful implementation than what is presented.
In the end, having to take the transitions (as opposed to teleporting between landfills) cost the route about 90 minutes, out of 1622, which is around a 5.5% difference. An improvement could only save us about that much. The obvious next step is to find a bound for the difference between teleportation and our output, however, due to time constraints, we leave it unanswered for now.

What we covered

1. The Problem
2. A Small Example
3. Real Data
4. Stars
5. Triangles
6. Route Building
7. Results
8. Using it
9. Conclusions/Future Work